**My Java Script Notes:**

1. **Explain the diffrerence in Var,let,const ?**

Till ES5, Java script makes use of **var** for assigning the variables. But in ES6, it introduces a new concept of **let** and **const** . The basis usage of let and const are similar to var, but still the scope of these are different.

Yes! For,

* var - Global Scope
* const- block scope, immutable
* let- block scope, mutable

Let see with an example,

1. var a;
2. a=12345;
3. console.log(a);
4. if(true){
5. let a=5;
6. console.log('inside if'+ a);
7. }
8. console.log('outside if'+ a);
10. //Output:
11. // 12345
12. // inside if 5
13. //outside if 12345

Hope you understand.

Similarly, for const I have mentioned that it has block scope and it is immutable. It is not completely immutable and also it is not completely re-assignable. Let see an example,

1. cont a = 5;
2. if(true)
3. {
4. a=10;
5. }
6. //Throws an error

The above example is not executable, It throws an error explaining that you are trying to assign a value for constant. It is immutable.

Whereas consider the following example,

1. const car = { type : 'diesel' };
2. if (true) {
3. car.type= 'petrol';
4. }
5. // Not completely immutable

This is not completely immutable. Meaning, you can reassign the object of the variable. But you are not allowed to reassign the variable itself. Have a look at the following snippet,

1. const car = { type: 'diesel'};
2. if (true) {
3. car = { type: 'petrol'};
4. }
6. //Throws an error

The above code will throws an error.

So, const is not completely immutable and at the same time it is not completely mutable too.

So finally, as [Daniel Sont](https://www.quora.com/profile/Daniel-Sont" \o "www.quora.com" \t "https://www.quora.com/_top) comment,  
“Use let all the time cause it looks pretty. Use const if you are paranoid. Use var only when let const-antly down.”

**Filter,Map and Reduce:**

const arr =[1,2,3,4,5,6,7];

const newArr = arr.filter(a => a % 2 ===0);

console.log(newArr);

VM303:3 (3) [2, 4, 6]

undefined

const arr =[1,2,3,4,5,6,7];

const newArr = arr.map(a => a % 2 === 0);

console.log(newArr);

VM343:1 Uncaught SyntaxError: Identifier 'arr' has already been declared

const arr1 =[1,2,3,4,5,6,7];

const newArr1 = arr1.map(a => a % 2 === 0);

console.log(newArr1);

VM370:3 (7) [false, true, false, true, false, true, false]

undefined

const arr2 =[1,2,3,4,5,6,7];

const newArr2 = arr2.map(a => a \*a === 0);

console.log(newArr2);

VM410:3 (7) [false, false, false, false, false, false, false]

undefined

const arr2 =[1,2,3,4,5,6,7];

const newArr2 = arr2.map(a => a \*a);

console.log(newArr2);

VM429:1 Uncaught SyntaxError: Identifier 'arr2' has already been declared

const arr3 =[1,2,3,4,5,6,7];

const newArr3 = arr3.map(a => a \*a);

console.log(newArr3);

VM452:3 (7) [1, 4, 9, 16, 25, 36, 49]

undefined

const arr4 =[1,2,3,4,5,6,7];

const newArr4 = arr4.map(a => a % 2);

console.log(newArr4);

VM498:3 (7) [1, 0, 1, 0, 1, 0, 1]

Undefined

AllCombined VeryImportant Program for Filter reduce and map.

**Demo.js:**

const ganesharr = [1,2,3,4,5,6,7];

const newGaneshArr = ganesharr

.filter(a => a % 2 ===0)

.map(a => a \* a)

.reduce((a,b) => a+b);

console.log(newGaneshArr);

// 56

**Realtime Program:**

const orders1 = [

{

userid : 1,

amount : 10

},

{

userid : 1,

amount : 15

},

{

userid : 2,

amount : 5

},

{

userid : 2,

amount : 100

}];

const result = orders1.filter(order => order.userid == 1)

.map(order => order.amount);

console.log(result);

//  [10, 15]

undefined

**Code 2 :**

const orders2 = [

{

userid : 1,

amount : 10

},

{

userid : 1,

amount : 15

},

{

userid : 2,

amount : 5

},

{

userid : 2,

amount : 100

}];

const result1 = orders2.filter(order => order.userid == 1)

.map(order => order.amount)

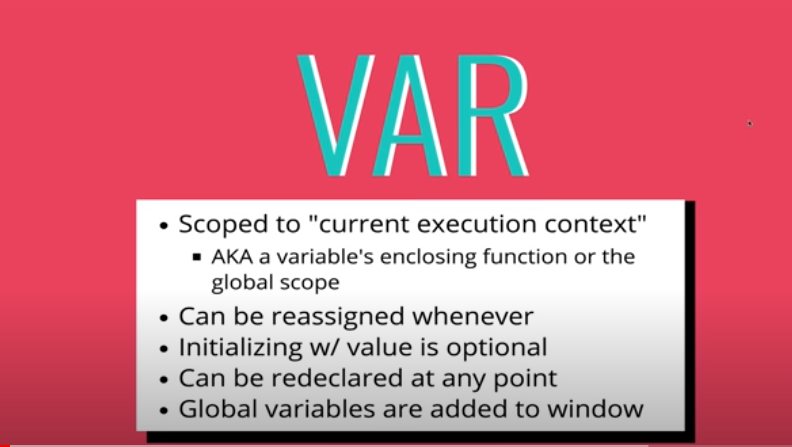
.reduce((a,b) => a+b);

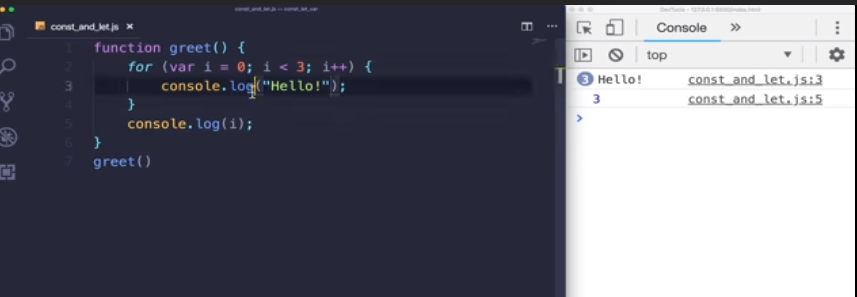
console.log(result1);

VM1763:21 25

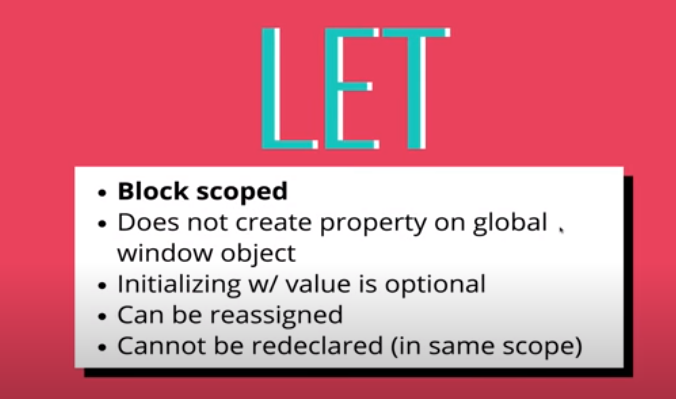
**Const ,let,var differences:**

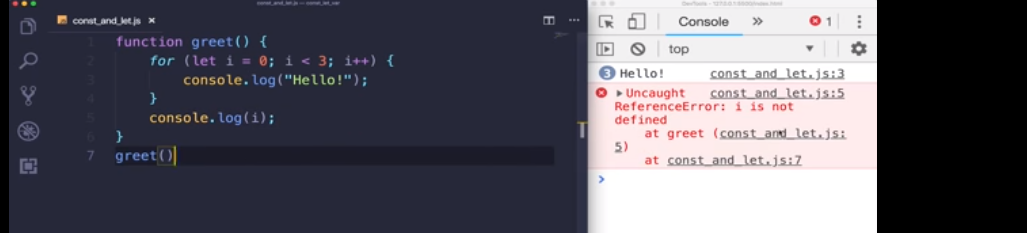
**Var:**





**Let:**



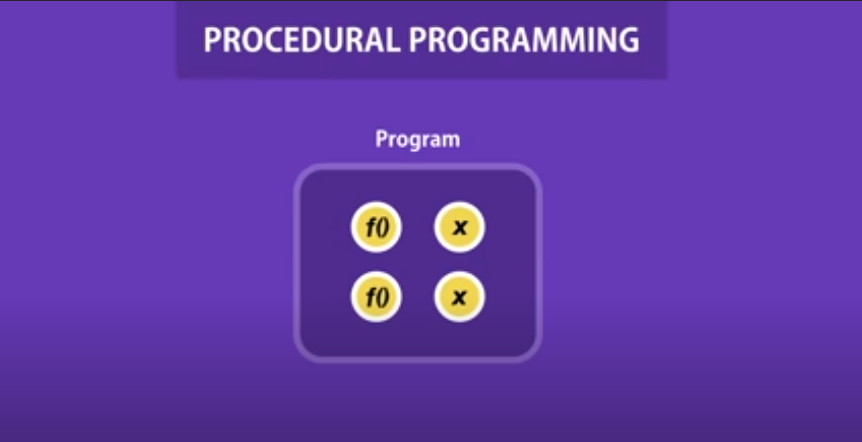


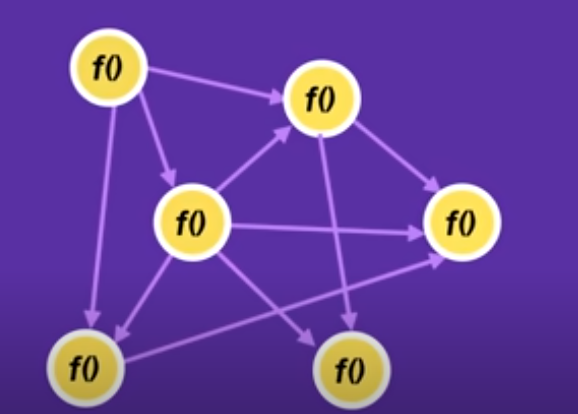
Const:

**Today(04/08/2020):**

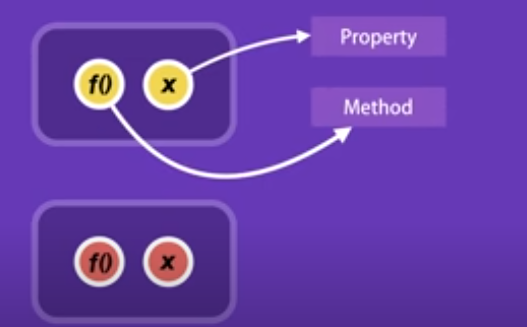
**Object Oriented Programming in Java Script:**

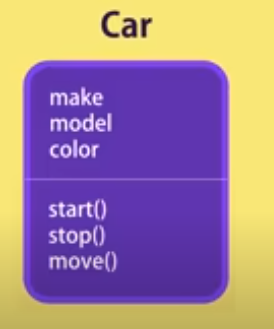






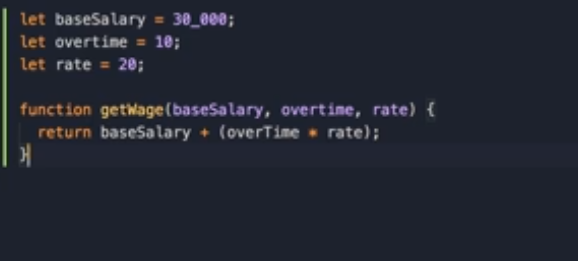




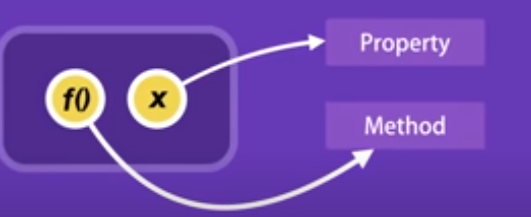


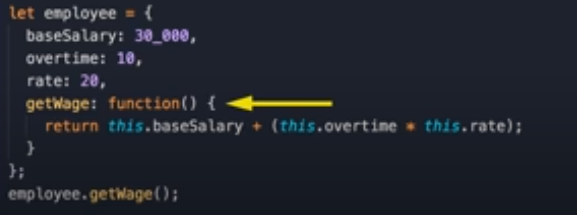
**Encapsulation:**

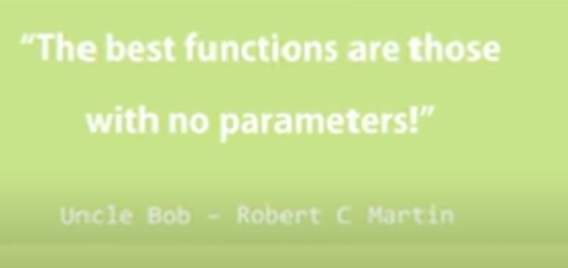
**Structured Programming**:



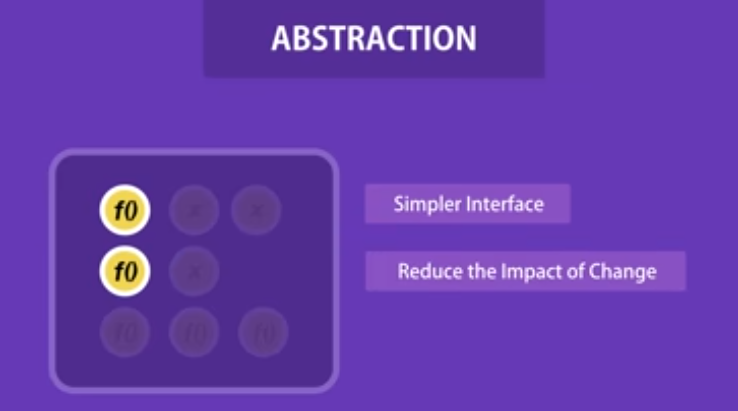
**Object oriented Programming:**





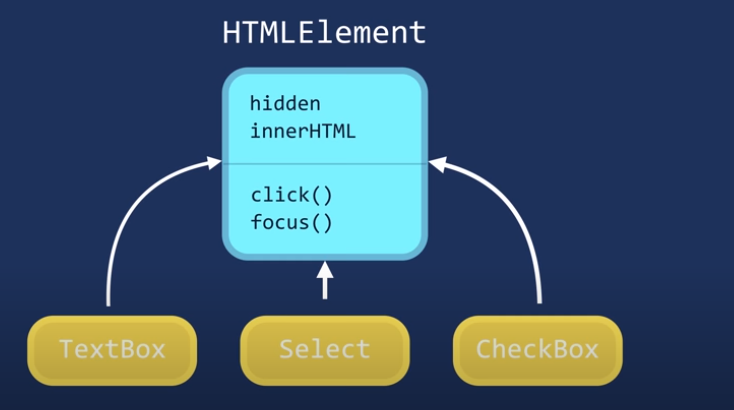


**Abstraction:**

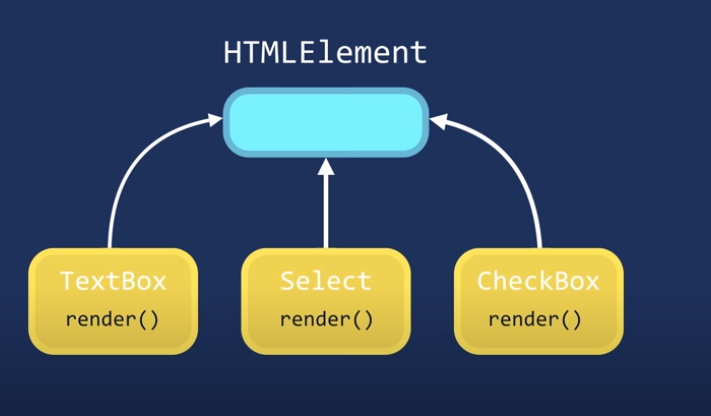


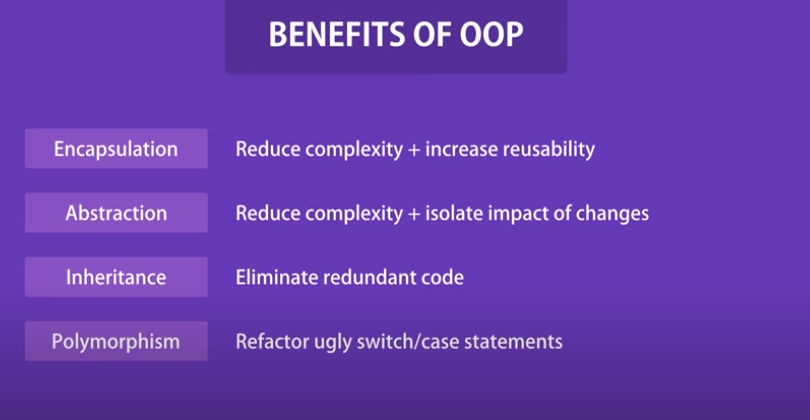
**Inheritance:**

To eleminate redundant code.



Polymorphism:



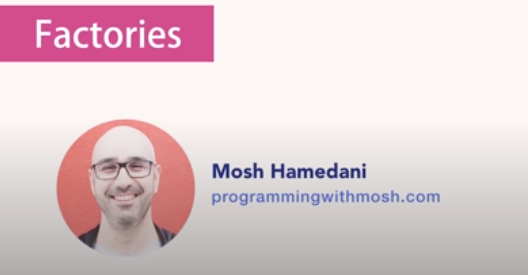


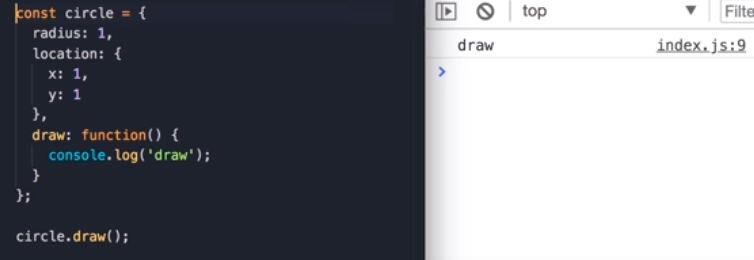


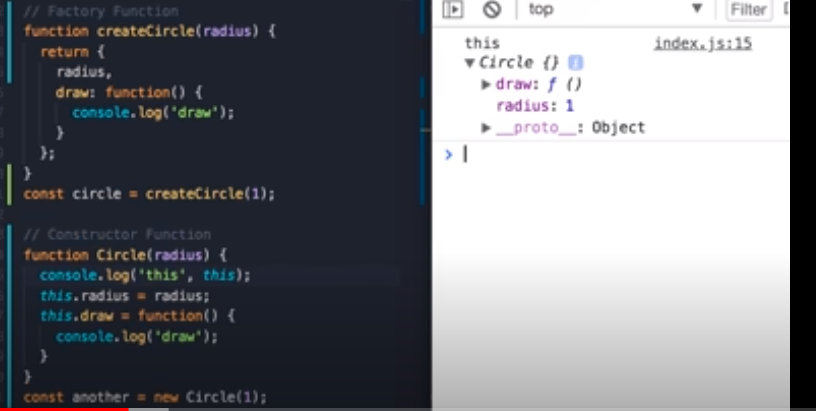




Example:

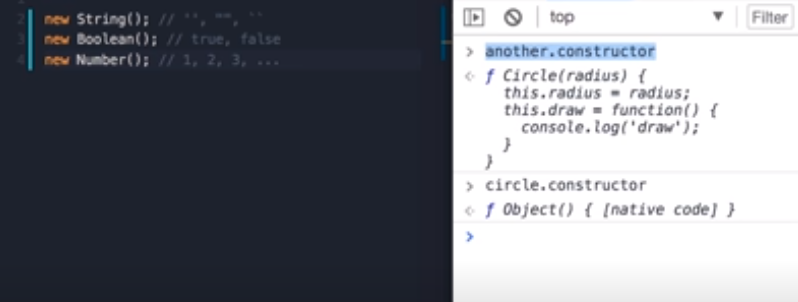












Examples: till now::

/\*

// creating an object in javascript

const circle =  {

    radius : 1,

    location: {

        x:1,

        y:1

    },

    draw: function() {

      console.log('draw');

    }

};

circle.draw(); \*/

//factory Function.

/\*function createCircle(radius) {

    return {

        radius,

        draw:function() {

            console.log('draw');

        }

    };

}

const circle = createCircle(1);

circle.draw();

\*/

// constructor Function

/\*funciton Circle(radius) {

    this.radius = radius;

    this.draw = funciton() {

        console.log('draw');

    }

}

const another = new Circle(1);

another.draw(); \*/

// constructor property

/\* let c ={};

let x = new Object();

new String(); // '', "", " "

new Boolean(); // true,false

new Number(); // 1,2,3,4......

\*/